A STATISTICAL TEXT-TO-PHONE FUNCTION USING NGRAMS AND RULES

William M. Fisher

National Institute of Standards and Technology (NIST)
Room A216, Building 225 (Technology)
Gaithersburg, MD 20899
william.fisher@nist.gov

ABSTRACT

Adopting concepts from statistical language modeling and rule-based transformations can lead to effective and efficient text-to-phone (TTP) functions. We present here the methods and results of one such effort, resulting in a relatively compact and fast set of TTP rules that achieves 94.5% segmental phonemic accuracy.

1. INTRODUCTION

Functions that transduce orthographic text into a phonemic or phonetic representation of its likely pronunciation find a number of uses in computer processing of speech and language. They are needed for synthesis from text, of course, but are also needed or useful for any symbolic computation that takes account of the sound of words. Examples include alignment of two text strings to maximize the sound similarity of corresponding words [6], the detection of splits and merges (in which a word in one text string corresponds to more than one word in another) [6], modeling speech recognizers to predict their performance [2]; and automatic generation of lexical entries for out-of-vocabulary (*OOV*) words for a speech recognizer [3] (the one this work was done to support).

Since the earliest days of TTP work, a popular approach has been to first look words up in a pronunciation dictionary or lexicon (a *pronlex*), applying a generative function – one that generates an output for *any* input, without dictionary look-up – only to those not found. The work reported here is concerned only with this dictionary-less generative function.

There have been a large number of efforts in the past to produce such general functions, both hand-crafted and data-driven, simple and complex [e.g. 1, 5, 7, 8, 9, 10, 11, 12]. This effort was directly inspired by the recent work of Meng et al. [9, 10, 11] and the now-standard Ngram methods for statistical language modeling [1].

2. GENERAL SCHEME

The basic concept of standard SLM work is expressed well by Stolcke [13, p. 270]: "An N-gram language model represents a probability distribution over words *w*, conditioned on (N-1)-tuples of preceding words, or histories *h*". Adapting this concept to TTP produces our basic *LnRm* model: a probability distribution over letter:phone mappings, conditioned on n-tuples of preceding characters and m-tuples of following characters.

While our initial work implemented this concept as indexed (n+m)-dimensional arrays, we now use the following string transformation rule formalism.

The logical form of the rules, common in Linguistics, is:

"T" => [P] / "L" __ "R"

where T (*target*), L (*left environment)*, and R (*right environment*) are character strings and P specifies which string of one or more phones is output. They are applied in one pass, transducing an input character string into a phonemic or phonetic representation.

The data structure holding the rules is a hash table whose key is a string representing the L, T, and R character strings of the rule, separated with ".", and whose content is P, specifying which phones correspond to the target letter string. This P specification is in general a probability distribution over phone strings, but for some applications, *including this one*, a rule may be determinized by reducing the P specification to just the single most likely phone string. For example, an L1R2 rule with probabilistic output and its deterministic version are:

KEY	CONTENTS
"b.a.th"	[/ae/,.80)(/ey/,.20)]
"b.a.th"	[/ae/]

We have so far dealt only with rules producing segmental lexical phonemes, viewing suprasegmentals and boundaries as the next problem to tackle. And while we have not done so yet, it would be easy to make inverse phone-to-text rules by simply interchanging the phone and letter strings in the training program. In our work so far, the target parts of each rule have been just one character wide, and so the *LnRm* terminology for describing the skeletal structure of the key part of a rule is sufficient. But our data structures can quite easily handle target parts of more than one letter, and we expect to adapt our logic for finding splits and merges [6] to produce "LnTxRm" rules in the future.

The basic step in estimating the probabilities is aligning character and phone exemplar strings and tallying a pot associated with each indicated correspondence. To produce the alignments, we use a slight adaptation of our general DP alignment procedure,

which finds an alignment that minimizes the sum of distance between the characters and phones put into correspondence. The atomic character:phone distances now come from an arbitrary table that we quickly put together. While this table is important to the whole algorithm, we didn't waste much effort on it because we have plans to develop an iterative flat-start procedure that will not need it.

3. EVALUATION METHODOLOGY

Making the assumption that the character-to-sound regularities found in new OOV words are typical of those found in general English allows us to evaluate TTP functions for this use by cross-validation using a pronlex for training and testing. An experiment consists of training the rules on a randomly selected fraction of the items in the pronlex and then testing them on another randomly-selected non-overlapping fraction of the items. Frequency weighting of lexical items is not used, since we have no basis for estimating the frequency of OOV words. The inaccuracy of the resulting function is expressed by the *phone error rate (PER)*, measured by applying NIST's standard alignment and error-finding scheme (as used in Sclite), treating the phone string from the pronlex as the REF symbol string and the phone string from the function as the HYP string. If a lexeme has several pronunciations, we count the lowest-PER one. Each data point we plot is the mean of 10 such randomized experiments.

In showing the power of methods such as ours, it is important to use a pronlex that is relatively free of error – especially non-systematic error. In order to reduce random error while not ending up with too small a pronlex, we used one that we call the "411" pronlex, which consists of only the word pronunciations that occur in a least 4 of 11 pronlexes we have on hand. This pool of pronlexes includes the LDC, CMU, UCLA, CELEX, TIMIT, and Moby ones, as well as several other proprietary ones. The 411 Pronlex contains 50,299 main items and 53,458 word pronunciations.

4. DETAILS

4.1 PER vs. Training

In order to get a feel for the effects of rule environment size (analogous to Ngram order) and amount of training on accuracy, we first did an experiment whose results are shown below as Figure 1. We measured and plotted the PER of several different types of trained-up rules as a function of different percentages of the pronlex used in training, always testing on a different randomly-selected 10% of the pronlex. The first four curves shown, labeled "px_lNrM", represent the results for sets of rules of type LnRm. No backoff was used in these first four. The general effects can be easily seen: the larger the rule environment (analogous to higher order of Ngram), the slower the error rate is reduced with more training, but the lower it gets asymptotically. The fifth curve, labeled "pxb_l1r2", shows the effect of backing off (to be discussed later), combining the rapid descent of the small-environment (low order) rules with the low ultimate error rate of the large-environment (high order) ones.

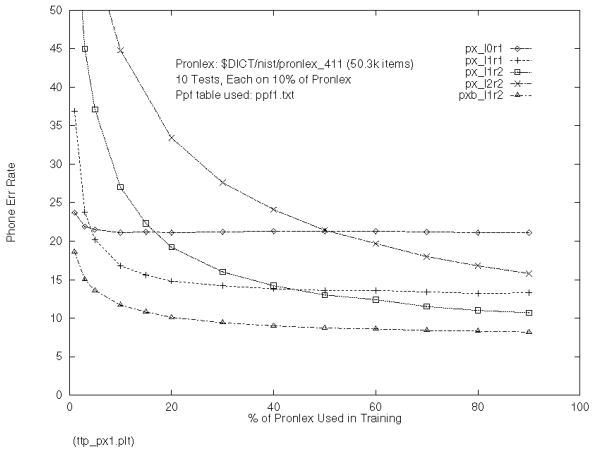

Fig. 1. Effect of Training on PER for Different Context Sizes.

4.2 Character Coverage vs. Training

Figure 2 below is a similar plot, showing the effects of rule environment size and amount of training on coverage of the input character N-graphs. Not surprisingly, as the environment size is increased, it takes more and more training data for the simple rules to achieve the same level of coverage. The last curve shown is for a backed-off version that, again not surprisingly, achieves total coverage with very little training.

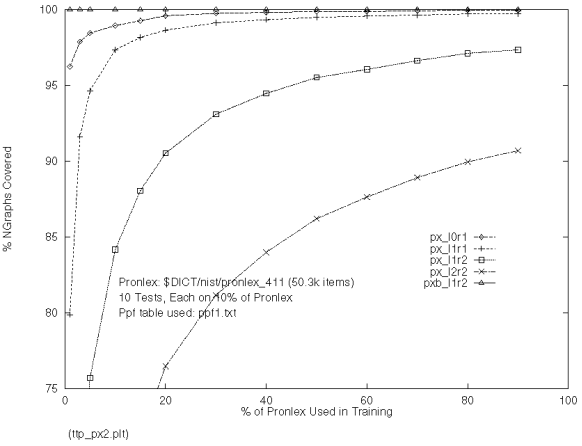

Fig. 2. Effect of Training on Letter Coverage for Different Context Sizes.

4.3 Backing Off

In conventional N-gram language modeling, the concept of *backing off* means that if the probability of a word cannot be estimated because the count of its N-gram is essentially zero, then one should back off to using the next lower order (N-1)-gram to estimate it, using an appropriate weight. Our implementation of this concept is, in general, to first check if

there is a rule in the ruleset matching the most specific key; and if not, to check for a less specific rule, and so on. For example, if we are transducing the string "bathe", with the cursor pointing at the "t", we would first check for the presence of a rule fitting the most specific schema, say "L3R3", which would be "#ba.t.he#". If our probe found no rule answering to such a key, we might back off to a schema of "L2R3", or "ba.t.he#". We assume that the likelihood of getting a hit on a rule increases with its generality. A *backoff path* is then a sequence of rule schemas to try when applying the rules.

In standard statistical language modeling, the succession to follow in backing off is not an issue: if there is no entry for a probability conditioned on the immediately preceding N words, then an entry conditioned on the immediately preceding N-1 words is sought. In our case, the action of the rule is conditioned by symbols both preceding and following the target, and the optimal backoff path is not obvious. For example, if we check and find no rule conditioned by two letters on the left and two on the right, should we next try one letter on the left and two on the right, or two on the left and one on the right? The question in general then is: What is the optimal backoff path to follow?

We have always assumed that the successor to the schema last tried should be one of more generality; since subparts consist of only literal strings, that implies reducing the number of symbols in the environment of the rule. Assuming further that the number of symbols should be decreased minimally (by 1) at each stage implies that the number of possible backoff paths starting from an LnRm rule is

$$npaths = (n+m)!$$

While this is a very large space, we have some preliminary evidence indicating that, at least for English, the proper generalization about the backoff path to follow is: if the current environment is unbalanced, back off to a balanced one; if it is balanced, back off to one with more symbols on the right. For example, the successor to "L2R3" would be "L2R2", and the successor to "L3R3" would be "L2R3".

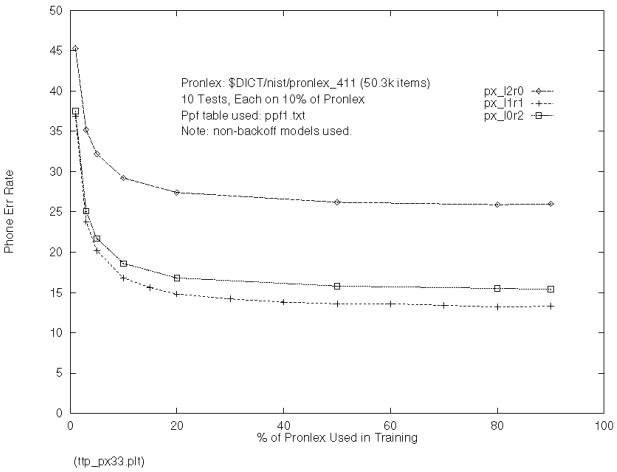

Fig. 3. Effect of Context Placement on PER.

Figure 3 above shows the preliminary evidence, being a plot of PER vs. size of training set for three different allocations of two environmental symbols to rule sub-parts, which can be viewed as three different backoff possibilities from the L1R2 schema. At least for this case, the generalization stated above holds.

4.4 Minimizing Ruleset Size

The space of possible rules, or alternatively array elements, is quite large: at least m**n, where m is the number of characters and n is the number of symbols in the conditioning environment. Our first programs, which used multi-dimensional arrays as the data structure, exhausted the memory on our largest computer when we tried to use more than five environmental symbols. Moving to the rule-based data structure, which can be viewed as a sparse-matrix technique, resulted in storing only 900k non-null values covering a space of at least 30**8 points in an experiment on L4R4 rules, a reduction by a factor of almost 10**6.

We have implemented another technique for minimizing the number of rules stored, which may have application back to the analogous arena of sentential language modeling: elimination of redundant rules. It is a special case of the general pruning rule of deleting that element whose deletion will increase the expected error rate the least.

Given a well-defined backoff path, each rule key A has a successor rule key B, in the sense that if rule key A is sought and not found, rule key B will be sought next. For example, given the partial backoff path (L2R2 --> L1R2), the successor of rule key "#b.a.th" is "b.a.th". Now if a particular rule set contains both a rule key A and its successor on the backoff trajectory, rule key B, and the output of both rules is identical, then we can delete the rule with key A without changing the input/output characteristics of the rule set. Figure 4 below shows us how much the size of the rule set produced by our standard procedure is decreased by such a strategy. At the largest rule environment tested, the reduction is from 900k rules to 20k, a factor of 45.

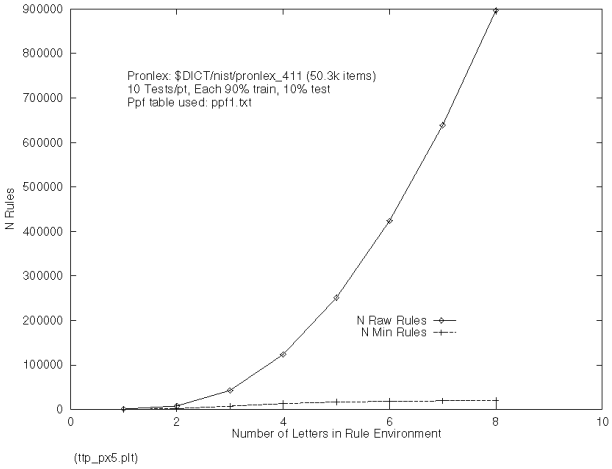

Fig. 4. Effect of Deleting Redundant Rules.

The cost of the strategy in terms of increased execution time is small, being a few more failed hash probes. It should be noted that this rule set has been determinized by the time we apply this pruning technique. Applying this strategy to non-deterministic cases where probabilities must be equal in order for the output of two rules to count as identical would certainly not pay off to the same degree.

5. FINAL RESULTS

Finally, to illustrate the high accuracy obtainable with this kind of algorithm, we present Figure 5, which is a plot of PER vs. the maximum number of letters in the environment of a rule in the rule set, analogous to the order N of an Ngram model. Our strategy for backing off is used, and minimization reduces the number of rules to fewer than 20k. The constant line shown for comparison is the most comparable result reported in Meng et al. 1996[11], although such comparisons should be taken with a grain of salt because the pronlexes, phoneme sets, and scoring algorithms used are at least somewhat different.

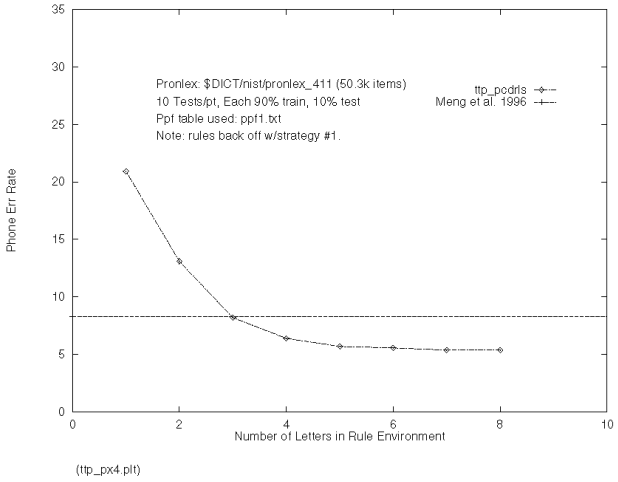

Fig. 5. Effect of Rule Environment Size on PER.

6. SUMMARY

We have shown that the simplest rule formalism, when coupled with the powerful algorithms of statistical training, hash coding, backing off, and deletion of redundant elements, can result in a compact and fast text-to-phone function that is highly accurate.

7. PROSPECTUS

There are a number of ways in which we plan to continue this work: 1) further experimentation to determine for sure the optimal back-off path; 2) trying to improve the rules by increasing the size of the target (T), using a version of our split/merge finding algorithm; and 3) post-processing the rules to yield fewer but more general ones.

8. REFERENCES

[1] Allen J., Hunnicutt S., and Klatt D. *From text to speech: the MITalk system*. MIT Press, Cambridge, MA, 1987.

[2] Chen, S., Beeferman, D., and Rosenfeld, R.. "Evaluation Metrics for Language Models". *Proceedings of the Broadcast News Transcription and Understanding Workshop*, sponsored by DARPA, pages 275-280, Morgan Kaufmann, 1998, ISBN 1-55860-564-9.

[3] Bahl, L.R., Das, S., de Souza, P.V., Epsten\in, M., Mercer, R.L., Merialdo, B., Nahamoo, D., Picheny, M.A., Powell, J.,. "Automatic Phonetic Baseform Determination". *IEEE International Conference on Acoustics, Speech, and Signal Processing.*, Toronto, May 1991, pages 173-176-62

[4] Clarkson, P., and Rosenfeld, R.. "Statistical Language Modeling Using the CMU-Cambridge Toolkit". *Proceedings ESCA EuroSpeech 1997*, Vol. 5, pages. 2707-2710, 1997.

[5] Elovitz, H.S., Johnson, R.W., McHugh, A., and Shore, J. *Automatic Translation of English Text to Phonetics by Means of Letter-to-Sound Rules*, NRL Report 7948, Naval Research Laboratory, Washington, D.C., 1976

[6] Fisher, W.M., and Fiscus, J.G.,. "Better Alignment Procedures for Speech Recognition Evaluation". *IEEE International Conference on Acoustics, Speech, and Signal Processing.*,Minneapolis, May 1993, Vol. II, pages 59-62.

[7] Hunnicutt, S. "Grapheme-to-Phoneme Rules: A Review". *Speech Transmission Laboratory Quarterly Progress and Status Report*, Royal Institute of Technology (KTH), Vol. STL-QPSR 2-3, pages 38-60, 1980.

[8] Luk, R.W.P., and Damper, R.I. ."Stochastic Phonographic Transduction for English", *Computer Speech and Language*, Vol. 10, pages 133-153, 1996.

[9] Meng, H., Seneff, S., and Zue, V."Phonological Parsing .for Reversible Letter-to-Sound/Sound-to-letter Generation", *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*,. Adelaide, April 1994, pages I-1-II-4.

[10] Meng, H. "*Phonological Parsing for Bi-directional Letter-to-Sound/Sound-to-Letter Generation*".. Ph.D. Thesis, M.I.T. Department of Electrical Engineering and Computer Science, Report MIT-LCS-TR-687,February 1995.

[11] Meng, H., Hunnicutt, S., Seneff, S., and Zue, V. "Reversible Letter-to-sound/Sound-to-letter Generation Based on Parsing Word Morpology [sic]". *Speech Communication*, Vol. 18, pages 47-63, 1996.

[12] Pagel, V., Lenzo, K., and Black, A.W. "Letter To Sound Rules for Accented Lexicon Compression", *Cmp-Lg Archive (URL http://xxx.lanl.gov/archive/cmp-lg)* paper #cmp-lg/9808010, August 1998.

[13] Stolcke, A.; "Entropy-based Pruning of Backoff Language Models". *Proceedings of the Broadcast News Transcription and Understanding Workshop, sponsored by DARPA*, pages 270-274, Morgan Kaufmann, 1998, ISBN 1-55860-564-9.